**१. Java भनेको के हो?**

Java एउटा **प्रोग्रामिङ भाषा** हो। जसले हामीलाई कम्प्युटरलाई भन्न दिन्छ, “यो काम गर” र कम्प्युटरले त्यो काम गर्छ।
यसलाई तपाईंले **कम्प्युटरसँग कुरा गर्ने भाषा** भनेर बुझ्न सक्नुहुन्छ।

**२. Java किन लोकप्रिय छ?**

* **सुरक्षित** हुन्छ — अरूलाई सजिलै ह्याक गर्न गाह्रो हुन्छ।
* **द्रुत** चल्छ।
* **मोबाइल, कम्प्युटर, रोबोट, गेम, वेबसाइट** जस्ता धेरै ठाउँमा चल्छ।
* “Write Once, Run Anywhere” — एक पटक लेखेपछि धेरै प्रकारको कम्प्युटरमा चलाउन सकिन्छ।

**३. Java कसरी काम गर्छ?**

Java प्रोग्राम लेखेर हामी **Java Compiler** नामको साधनलाई दिन्छौं।
यो त्यसलाई **Bytecode** मा बदल्छ।
Bytecode कम्प्युटरमा **Java Virtual Machine (JVM)** मार्फत चल्छ।

**सरल उदाहरण:**
यो त्यस्तै हो, तपाईं नेपालीमा लेख्नुहुन्छ → अनुवादकले अंग्रेजीमा बदल्छ → सबैले बुझ्छन्।

**४. Java सुरु गर्न के चाहिन्छ?**

1. **Java Development Kit (JDK)** — Java प्रोग्राम बनाउन र चलाउन।
2. **IDE (जस्तै IntelliJ, Eclipse, NetBeans)** — कोड लेख्न सजिलो बनाउने सफ्टवेयर।
3. कम्प्युटर (Windows, Mac, वा Linux)।

**५. सबैभन्दा साधारण Java प्रोग्राम**

java

CopyEdit

public class HelloWorld {

 public static void main(String[] args) {

 System.out.println("Namaste Sansar");

 }

}

* **public class HelloWorld** — प्रोग्रामको नाम।
* **main** — यहाँबाट प्रोग्राम सुरु हुन्छ।
* **System.out.println** — स्क्रिनमा कुरा देखाउने आदेश।

**६. Java का मुख्य भागहरू (Building Blocks)**

* **Variables (परिवर्तनीय)** — डाटा राख्ने बाकस।
* **Data Types** — कुन प्रकारको डाटा हो (जस्तै: number, text)।
* **Operators** — गणितीय वा तार्किक काम गर्ने चिन्ह।
* **Control Statements** — if, else, loops जस्ता, जुनले प्रोग्रामको बाटो बदल्छ।
* **Methods** — आदेशहरूको समूह।
* **Classes & Objects** — Java को मुख्य आधार, जसले कोडलाई व्यवस्थित बनाउँछ।

**७. किन सिक्ने?**

* भविष्यमा **प्रोग्रामर** बन्न।
* मोबाइल एप बनाउने।
* गेम बनाउने।
* रोबोट वा AI बनाउन।

**Chapter 1 – Java सुरु गर्ने पहिलो कदम**

**१. तयारी**

Java कोड लेख्नका लागि तपाईंलाई यी चाहिन्छ:

1. **Java Development Kit (JDK)** — Java प्रोग्राम बनाउन र चलाउन।
2. **IDE वा Code Editor** — जस्तै *IntelliJ IDEA*, *Eclipse*, वा *VS Code*।
3. तपाईंको कम्प्युटरमा **JDK install** भएको हुनुपर्छ।

सानो टिप: यदि तपाईंलाई JDK कसरी download र install गर्ने थाहा छैन भने, म step-by-step नेपालीमा चित्रसहित पनि दिन सक्छु।

**२. आफ्नो पहिलो Java प्रोग्राम**

तपाईंले Java मा सबैभन्दा पहिले "Hello World" प्रोग्राम लेख्नुहुन्छ।
यसको अर्थ कम्प्युटरलाई भन्नु — "यो शब्द देखाउ।"

java

CopyEdit

public class MyFirstProgram {

 public static void main(String[] args) {

 System.out.println("Namaste Sansar");

 }

}

**३. यो कोडको व्याख्या**

* **public class MyFirstProgram**
यो तपाईंको प्रोग्रामको नाम हो। *class* भनेको कोड राख्ने बाकस हो।
* **public static void main(String[] args)**
यहाँबाट कम्प्युटरले प्रोग्राम सुरु गर्छ। यो भनेको “यो बाटो बाट सुरु गर” भन्ने जस्तो हो।
* **System.out.println("Namaste Sansar");**
स्क्रिनमा "Namaste Sansar" देखाउने आदेश।

**४. कोड चलाउने तरिका**

1. यो कोड **MyFirstProgram.java** नामले save गर्नुहोस्।
2. टर्मिनल वा IDE मा compile गर्नुहोस्:

nginx

CopyEdit

javac MyFirstProgram.java

1. त्यसपछि चलाउनुहोस्:

nginx

CopyEdit

java MyFirstProgram

1. स्क्रिनमा तपाईं देख्नुहुनेछ:

nginx

CopyEdit

Namaste Sansar

**५. अभ्यास कार्य**

**कार्य १:**
"Namaste Sansar" लाई बदल्नुहोस् र आफ्नो नाम देखाउने प्रोग्राम बनाउनुहोस्।
उदाहरण:

java

CopyEdit

System.out.println("Mero Naam Lok Ho");

**कार्य २:**
एकै पटक दुई कुरा देखाउने कोड लेख्नुहोस्:

java

CopyEdit

System.out.println("Namaste!");

System.out.println("Tapai kasto hunuhunchha?");

**Chapter 2 – Variables, Data Types र गणित**

**१. Variable भनेको के हो?**

Variable भनेको **डाटा राख्ने बाकस** हो, जसमा तपाईं केही जानकारी (जस्तै संख्या, शब्द) राख्न सक्नुहुन्छ।
यो बाकसलाई तपाईं नाम दिनुहुन्छ, अनि पछि प्रयोग गर्नुहुन्छ।

उदाहरण:

java

CopyEdit

int umar = 12; // उमेर

String naam = "Lok"; // नाम

**२. Java का Common Data Types**

Java मा धेरै प्रकारका Data Types छन्, तर अहिले हामीले सामान्य सिक्नेछौं:

| **Data Type** | **अर्थ** | **उदाहरण** |
| --- | --- | --- |
| int | पुरा संख्या | 5, 100, -7 |
| double | दशमलव संख्या | 5.5, 3.14 |
| String | शब्द/वाक्य | "Namaste", "Hello" |
| boolean | True/False | true, false |

**३. Variables प्रयोग गर्ने Example**

java

CopyEdit

public class VariableExample {

 public static void main(String[] args) {

 int umar = 12;

 String naam = "Lok";

 double mathMarks = 88.5;

 boolean pass = true;

 System.out.println("Mero naam: " + naam);

 System.out.println("Mero umar: " + umar);

 System.out.println("Math Marks: " + mathMarks);

 System.out.println("Pass Gareko: " + pass);

 }

}

📌 यहाँ + चिन्हले String र Variable जोड्छ।

**४. गणित (Math) Java मा**

Java मा तपाईं सजिलै गणित गर्न सक्नुहुन्छ:

* + — जोड
* - — घटाउ
* \* — गुणा
* / — भाग
* % — बाकी (remainder)

उदाहरण:

java

CopyEdit

public class MathExample {

 public static void main(String[] args) {

 int a = 10;

 int b = 3;

 System.out.println("Jod: " + (a + b)); // 13

 System.out.println("Ghat: " + (a - b)); // 7

 System.out.println("Guna: " + (a \* b)); // 30

 System.out.println("Bhaag: " + (a / b)); // 3

 System.out.println("Baki: " + (a % b)); // 1

 }

}

**५. अभ्यास कार्य**

**कार्य १:**
आफ्नो नाम, उमेर, र पढाइको स्तर देखाउने Java प्रोग्राम लेख्नुहोस्।

**कार्य २:**
दुई संख्याको **जोड, घटाउ, गुणा, भाग** देखाउने प्रोग्राम बनाउनुहोस्।

**कार्य ३:**
double प्रकार प्रयोग गरेर दशमलव संख्यामा गणित गर्नुहोस् (जस्तै 5.5 + 2.3)।

**Chapter 3 – If-Else र निर्णय लिन सक्ने प्रोग्राम**

**१. If-Else भनेको के हो?**

If-Else भनेको कम्प्युटरलाई भन्नु —
"यदि (if) यो कुरा सच्छ हो भने यस्तो गर,
अथवा (else) अरू कुरा गर।"

उदाहरण:
यदि मौसममा घाम छ भने "पिकनिक जाऔं", नभए "घरमै बसौं।"

**२. Java मा If-Else को संरचना**

java

CopyEdit

if (condition) {

 // condition सच्छ (true) भए यो चल्छ

} else {

 // condition झुटो (false) भए यो चल्छ

}

**३. साधारण उदाहरण**

java

CopyEdit

public class IfElseExample {

 public static void main(String[] args) {

 int umar = 15;

 if (umar >= 18) {

 System.out.println("Tapai vote din saknuhunchha.");

 } else {

 System.out.println("Tapai vote din saknuhunna.");

 }

 }

}

📌 यहाँ:

* umar >= 18 — Condition हो (१८ वर्ष वा बढी भए मात्र vote दिन पाइन्छ)।
* >= — “बढी वा बराबर” भन्ने चिन्ह।

**४. If-Else If-Else**

एकै पटक धेरै condition जाँच्नका लागि:

java

CopyEdit

public class GradeExample {

 public static void main(String[] args) {

 int marks = 85;

 if (marks >= 90) {

 System.out.println("Grade: A+");

 } else if (marks >= 75) {

 System.out.println("Grade: A");

 } else if (marks >= 50) {

 System.out.println("Grade: B");

 } else {

 System.out.println("Fail");

 }

 }

}

**५. Comparison Operators (तुलना गर्ने चिन्ह)**

| **चिन्ह** | **अर्थ** |
| --- | --- |
| == | बराबर |
| != | बराबर होइन |
| > | ठूलो |
| < | सानो |
| >= | ठूलो वा बराबर |
| <= | सानो वा बराबर |

**६. अभ्यास कार्य**

**कार्य १:**
एक प्रोग्राम बनाउनुहोस् जसले उमेर चेक गरेर भन्न सकोस् —
"बालबालिका", "किशोर", वा "वयस्क"।

**कार्य २:**
दुई संख्यामा कुन ठूलो छ भनेर देखाउने प्रोग्राम बनाउनुहोस्।

**कार्य ३:**
विद्यार्थीको अंक अनुसार Grade देखाउने (A+, A, B, Fail) प्रोग्राम बनाउनुहोस्।

**Chapter 4 – Loops (For, While, Do-While)**

**१. Loop भनेको के हो?**

Loop भनेको कम्प्युटरलाई आदेश दिने तरिका हो —
"यो काम फेरि-फेरि गर, जबसम्म एउटा शर्त (condition) पूरा हुँदैन।"

**२. Loop का प्रकार**

Java मा मुख्यत: ३ प्रकारका loops हुन्छन्:

1. **for loop**
2. **while loop**
3. **do-while loop**

**1. For Loop**

यो त्यस्तो loop हो जहाँ हामीले सुरु, अन्त्य र बढाउने नियम सबै सुरुमा नै लेख्छौं।

संरचना:

java

CopyEdit

for (सुरु; condition; बढाउने/घटाउने) {

 // दोहोर्याएर गर्ने काम

}

उदाहरण:

java

CopyEdit

public class ForLoopExample {

 public static void main(String[] args) {

 for (int i = 1; i <= 5; i++) {

 System.out.println("Number: " + i);

 }

 }

}

📌 यहाँ:

* int i = 1 → सुरुवात 1 बाट।
* i <= 5 → जबसम्म i 5 भन्दा सानो वा बराबर छ।
* i++ → हरेक पटक 1 बढाउ।

**2. While Loop**

यो loop तबसम्म चल्छ, जबसम्म condition **true** छ।

संरचना:

java

CopyEdit

while (condition) {

 // दोहोर्याएर गर्ने काम

}

उदाहरण:

java

CopyEdit

public class WhileLoopExample {

 public static void main(String[] args) {

 int i = 1;

 while (i <= 5) {

 System.out.println("Number: " + i);

 i++;

 }

 }

}

**3. Do-While Loop**

यो while loop जस्तै हो, तर **कम्तीमा एक पटक** चल्छ, चाहे condition false भए पनि।

संरचना:

java

CopyEdit

do {

 // दोहोर्याएर गर्ने काम

} while (condition);

उदाहरण:

java

CopyEdit

public class DoWhileExample {

 public static void main(String[] args) {

 int i = 1;

 do {

 System.out.println("Number: " + i);

 i++;

 } while (i <= 5);

 }

}

**४. अभ्यास कार्य**

**कार्य १:**
For loop प्रयोग गरेर १ देखि १० सम्मका संख्या देखाउने प्रोग्राम बनाउनुहोस्।

**कार्य २:**
While loop प्रयोग गरेर २ को table (२, ४, ६ … २०) देखाउने प्रोग्राम बनाउनुहोस्।

**कार्य ३:**
Do-While प्रयोग गरेर १ देखि ५ सम्मका संख्या देखाउने प्रोग्राम बनाउनुहोस्।

**Chapter 5 – Arrays (एर्रे)**

**१. Array भनेको के हो?**

Array भनेको **एउटा लामो बाकस** हो, जसमा धेरै समान प्रकारका डेटा क्रम अनुसार राखिन्छन्।
यो त्यस्तै हो जस्तो १० वटा विद्यार्थीको नाम राख्न १० वटा अलग-अलग variable बनाउने सट्टा **एउटा array** प्रयोग गर्ने।

**२. Array किन प्रयोग गर्ने?**

* धेरै डेटा एकै ठाउँमा राख्न।
* Index (क्रम संख्या) मार्फत सजिलै पहुँच गर्न।
* कोड छोटो र सजिलो बनाउन।

**३. Array कसरी बनाउने?**

Java मा array बनाउनको संरचना:

java

CopyEdit

dataType[] arrayName = new dataType[size];

वा

java

CopyEdit

dataType[] arrayName = {value1, value2, value3};

**४. उदाहरण**

java

CopyEdit

public class ArrayExample {

 public static void main(String[] args) {

 String[] students = {"Ram", "Sita", "Hari", "Gita"};

 // पहिलो नाम देखाउने

 System.out.println("Pahilo Student: " + students[0]);

 // सबै नाम देखाउने

 for (int i = 0; i < students.length; i++) {

 System.out.println("Student " + (i + 1) + ": " + students[i]);

 }

 }

}

📌 यहाँ:

* students[0] → पहिलो नाम।
* students.length → array को लम्बाइ (कति डेटा छन्)।

**५. संख्याको Array**

java

CopyEdit

public class NumberArrayExample {

 public static void main(String[] args) {

 int[] marks = {85, 90, 78, 92};

 int total = 0;

 for (int i = 0; i < marks.length; i++) {

 total = total + marks[i];

 }

 System.out.println("Total Marks: " + total);

 System.out.println("Average: " + (total / marks.length));

 }

}

**६. अभ्यास कार्य**

**कार्य १:**
५ वटा मनपर्ने खानेकुराको नाम array मा राखेर देखाउने प्रोग्राम बनाउनुहोस्।

**कार्य २:**
५ वटा अंक राखेर त्यसको कुल र औसत निकाल्ने प्रोग्राम बनाउनुहोस्।

**कार्य ३:**
१ देखि १० सम्मका संख्याको table array मा राखेर सबै देखाउने प्रोग्राम बनाउनुहोस्।

**Chapter 6 – Methods (फङ्सन)**

**१. Method भनेको के हो?**

Method भनेको **कोडको एउटा समूह** हो जसलाई नाम दिइन्छ, ताकि पछि त्यसलाई सजिलै बोलाएर (call गरेर) प्रयोग गर्न सकिन्छ।
यो त्यस्तै हो —
एक पटक "चिया बनाउने तरिका" लेखिदियो, अनि जब जब चिया चाहियो, केवल *"चिया बनाऊ"* भन्नु मात्र पर्छ।

**२. Methods किन प्रयोग गर्ने?**

* कोड **सफासुथर** हुन्छ।
* एउटै काम फेरि-फेरि गर्न सकिन्छ, कोड दोहोर्याएर नलेखिकन।
* सजिलो Debugging (त्रुटि खोज्न)।

**३. Method को संरचना**

java

CopyEdit

returnType methodName(parameters) {

 // काम गर्ने कोड

 return value; // यदि returnType void होइन भने

}

**४. साधारण Method उदाहरण**

java

CopyEdit

public class MethodExample {

 // Method बनाउने

 public static void greet() {

 System.out.println("Namaste!");

 }

 public static void main(String[] args) {

 // Method call गर्ने

 greet();

 greet();

 }

}

📌 यहाँ:

* public static void greet() — Method को नाम greet हो, void मतलब केही फर्काउँदैन।
* greet(); — Method call गर्न।

**५. Parameter सहितको Method**

java

CopyEdit

public class MethodWithParameter {

 public static void sayHello(String name) {

 System.out.println("Namaste " + name + "!");

 }

 public static void main(String[] args) {

 sayHello("Ram");

 sayHello("Sita");

 }

}

📌 यहाँ:

* Method ले String name इनपुट लिन्छ।
* call गर्दा नाम दिनुपर्छ — sayHello("Ram");

**६. Value फर्काउने Method**

java

CopyEdit

public class MethodReturnExample {

 public static int addNumbers(int a, int b) {

 return a + b;

 }

 public static void main(String[] args) {

 int sum = addNumbers(5, 7);

 System.out.println("Jod: " + sum);

 }

}

📌 यहाँ:

* Method को **return type** int छ।
* return a + b; → method ले संख्या फर्काउँछ।

**७. अभ्यास कार्य**

**कार्य १:**
एक method बनाउनुहोस् जसले तपाईंको नाम देखाओस्, अनि main method बाट तीन पटक call गर्नुहोस्।

**कार्य २:**
एक method बनाउनुहोस् जसले दुई संख्या लिई jod, ghat, guna, bhaag फर्काओस्।

**कार्य ३:**
एक method बनाउनुहोस् जसले string इनपुट लिई त्यसलाई उल्टो (reverse) गरेर फर्काओस्।

**Chapter 7 – Class र Object**

**१. Class भनेको के हो?**

Class भनेको **ब्लूप्रिन्ट (नक्सा)** हो जसबाट object बनाइन्छ।
यो त्यस्तै हो — कार बनाउनको लागि blueprint बनाउने, अनि blueprint बाट धेरै कार बनाउन सकिन्छ।

Class भित्र:

* Variable (डेटा राख्ने)
* Methods (काम गर्ने तरिका)

**२. Object भनेको के हो?**

Object भनेको **class को वास्तविक रूप** हो।
Class केवल योजना हो, तर object वास्तविक चीज हो जसले काम गर्छ।

उदाहरण:

* Class = “मान्छे” को design
* Object = Ram, Sita, Hari (अलग मान्छेहरू)

**३. Class र Object को उदाहरण**

java

CopyEdit

// Class बनाउने

public class Person {

 String name;

 int age;

 void sayHello() {

 System.out.println("Namaste, mero naam " + name + " ho.");

 }

}

// मुख्य प्रोग्राम

public class Main {

 public static void main(String[] args) {

 // Object बनाउने

 Person p1 = new Person();

 p1.name = "Ram";

 p1.age = 20;

 p1.sayHello();

 Person p2 = new Person();

 p2.name = "Sita";

 p2.age = 18;

 p2.sayHello();

 }

}

📌 यहाँ:

* Person — Class
* p1, p2 — Object
* name र age — Properties (Variable)
* sayHello() — Method

**४. Constructor**

Constructor भनेको **object बनाउँदा automatic चल्ने method** हो।
यसले object बनाउँदा प्रारम्भिक मान दिन सजिलो बनाउँछ।

java

CopyEdit

public class Person {

 String name;

 int age;

 // Constructor

 Person(String n, int a) {

 name = n;

 age = a;

 }

 void sayHello() {

 System.out.println("Namaste, mero naam " + name + " ho. Umar: " + age);

 }

}

public class Main {

 public static void main(String[] args) {

 Person p1 = new Person("Ram", 20);

 Person p2 = new Person("Sita", 18);

 p1.sayHello();

 p2.sayHello();

 }

}

**५. अभ्यास कार्य**

**कार्य १:**
Car नामको class बनाउनुहोस् जसमा:

* Properties: brand, model, price
* Method: कारको जानकारी देखाउने

**कार्य २:**
Student नामको class बनाउनुहोस् जसमा:

* Properties: name, rollNo, marks
* Method: पास/फेल देखाउने (marks >= 40 पास)

**कार्य ३:**
Constructor प्रयोग गरेर आफ्नो मनपर्ने object बनाउनुहोस् र त्यसको जानकारी देखाउनुहोस्।

**Chapter 8 – Inheritance (विरासत) र Polymorphism (बहुरूपता)**

**१. Inheritance भनेको के हो?**

Inheritance भनेको **एउटा class को गुण (properties) र व्यवहार (methods) अर्को class ले पाउनु** हो।
यो त्यस्तै हो —
**बाबु-आमाबाट छोराछोरीले गुण पाउने।**

**२. Inheritance किन प्रयोग गर्ने?**

* कोड दोहोर्याएर नलेख्न।
* पुरानो कोड पुन: प्रयोग गर्न।
* सम्बन्धित class हरू बीच hierarchy बनाउन।

**३. Inheritance को संरचना**

Java मा inheritance बनाउनको लागि extends शब्द प्रयोग हुन्छ।

java

CopyEdit

class Parent {

 void greet() {

 System.out.println("Namaste! Ma Parent ho.");

 }

}

class Child extends Parent {

 void introduce() {

 System.out.println("Ma Child ho.");

 }

}

public class Main {

 public static void main(String[] args) {

 Child c = new Child();

 c.greet(); // Parent class को method

 c.introduce(); // Child class को method

 }

}

📌 यहाँ:

* Child extends Parent → Child ले Parent का method र properties पायो।
* अब Child object ले Parent को method पनि प्रयोग गर्न सक्छ।

**४. Types of Inheritance (Java मा)**

* **Single Inheritance** — एउटा parent बाट एउटा child।
* **Multilevel Inheritance** — Parent → Child → Grandchild।
* **Hierarchical Inheritance** — एउटै parent बाट धेरै child।

Java मा multiple inheritance (एक child का धेरै parent) **direct support हुँदैन** (interfaces मार्फत मात्र हुन्छ)।

**Polymorphism (बहुरूपता)**

**५. Polymorphism भनेको के हो?**

Polymorphism भनेको **एउटै नामको method, तर फरक-फरक तरिकाले काम गर्ने क्षमता** हो।
यो दुई प्रकारको हुन्छ:

1. **Compile-time Polymorphism (Method Overloading)**
2. **Runtime Polymorphism (Method Overriding)**

**६. Method Overloading (Compile-time)**

एउटै method नाम, तर parameter फरक।

java

CopyEdit

class MathOperation {

 int add(int a, int b) {

 return a + b;

 }

 double add(double a, double b) {

 return a + b;

 }

}

public class Main {

 public static void main(String[] args) {

 MathOperation m = new MathOperation();

 System.out.println(m.add(5, 7)); // int version

 System.out.println(m.add(5.5, 2.3)); // double version

 }

}

**७. Method Overriding (Runtime)**

Parent को method लाई child ले आफ्नो तरिकाले पुन: लेख्ने।

java

CopyEdit

class Animal {

 void sound() {

 System.out.println("Some sound...");

 }

}

class Dog extends Animal {

 @Override

 void sound() {

 System.out.println("Bark!");

 }

}

public class Main {

 public static void main(String[] args) {

 Animal a = new Dog(); // Parent reference, Child object

 a.sound(); // "Bark!"

 }

}

📌 यहाँ:

* @Override → पुरानो method लाई नयाँ version दिइयो।
* Runtime मा कुन method चल्ने भन्ने निर्णय हुन्छ।

**८. अभ्यास कार्य**

**कार्य १:**
Vehicle class र Car class बनाउनुहोस् जसमा Car ले Vehicle बाट गुण पाओस्।

**कार्य २:**
Calculator class मा add method लाई तीन version मा बनाउनुहोस् (२ int, २ double, ३ int)।

**कार्य ३:**
Animal class र तीन child (Dog, Cat, Cow) बनाएर sound() method override गर्नुहोस्।

**Chapter 9 – Encapsulation (घेराबन्दी) र Abstraction (अवधारणा)**

**१. Encapsulation भनेको के हो?**

Encapsulation भनेको **डेटा (variables) र त्यसमा काम गर्ने method लाई एउटै class भित्र राख्ने तर direct बाहिरबाट पहुँच नदिनु** हो।
यो *जस्तो बैंकको locker — तपाईं बाहिरबाट सीधा पहुँच पाउनुहुन्न, तर अधिकृत तरिकाले मात्र खोल्न सकिन्छ।*

**२. Encapsulation किन प्रयोग गर्ने?**

* डेटा **सुरक्षित** राख्न।
* बाहिरबाट direct change हुन नदिन।
* Controlled access दिन (getter/setter methods मार्फत)।

**३. Encapsulation को उदाहरण**

java

CopyEdit

class BankAccount {

 private double balance; // direct बाहिरबाट पहुँच हुँदैन

 // Constructor

 BankAccount(double initialBalance) {

 balance = initialBalance;

 }

 // Getter

 public double getBalance() {

 return balance;

 }

 // Setter

 public void deposit(double amount) {

 if (amount > 0) {

 balance += amount;

 }

 }

 public void withdraw(double amount) {

 if (amount > 0 && amount <= balance) {

 balance -= amount;

 }

 }

}

public class Main {

 public static void main(String[] args) {

 BankAccount acc = new BankAccount(5000);

 acc.deposit(2000);

 acc.withdraw(1000);

 System.out.println("Balance: " + acc.getBalance());

 }

}

📌 यहाँ:

* private → variable बाहिरबाट direct access हुँदैन।
* getBalance() → पढ्न।
* deposit() र withdraw() → परिवर्तन गर्न, तर नियम अनुसार मात्र।

**Abstraction (अवधारणा)**

**४. Abstraction भनेको के हो?**

Abstraction भनेको **काम कसरी हुन्छ भन्ने विवरण लुकाएर केवल प्रयोग गर्ने तरिका देखाउने** हो।
यो *जस्तो टिभीको remote — तपाईंले कुन बटनले के गर्छ थाहा हुन्छ, तर भित्र circuitry कसरी चल्छ थाहा हुनुपर्दैन।*

**५. Abstraction किन प्रयोग गर्ने?**

* कोड **सफा** र सजिलो बनाउन।
* जटिल विवरण लुकाउन।
* केवल आवश्यक कुरा मात्र देखाउन।

**६. Abstract Class को उदाहरण**

java

CopyEdit

abstract class Animal {

 abstract void sound(); // method को body छैन

 void eat() {

 System.out.println("Eating...");

 }

}

class Dog extends Animal {

 @Override

 void sound() {

 System.out.println("Bark!");

 }

}

public class Main {

 public static void main(String[] args) {

 Animal a = new Dog();

 a.sound(); // "Bark!"

 a.eat(); // "Eating..."

 }

}

📌 यहाँ:

* abstract class → जसबाट object direct बनाउन सकिँदैन।
* abstract method → body छैन, child class ले define गर्नुपर्छ।
* Non-abstract method पनि राख्न सकिन्छ।

**७. Interface प्रयोग गरेर Abstraction**

java

CopyEdit

interface Vehicle {

 void start();

 void stop();

}

class Car implements Vehicle {

 public void start() {

 System.out.println("Car Started");

 }

 public void stop() {

 System.out.println("Car Stopped");

 }

}

public class Main {

 public static void main(String[] args) {

 Vehicle v = new Car();

 v.start();

 v.stop();

 }

}

📌 यहाँ:

* interface → सबै methods abstract हुन्छन्।
* implements → interface को सबै methods define गर्नुपर्छ।

**८. अभ्यास कार्य**

**कार्य १:**
Encapsulation प्रयोग गरेर Student class बनाउनुहोस् जसमा नाम, उमेर, र grade राख्न सकियोस्, र getter/setter मार्फत मात्र पहुँच पाओस्।

**कार्य २:**
Abstract class Shape बनाउनुहोस् जसमा area() method होस्, अनि Circle र Rectangle class ले त्यसलाई override गरोस्।

**कार्य ३:**
Interface Playable बनाउनुहोस् जसमा play() method होस्, र Football र Cricket class ले त्यसलाई implement